PALGO JOURNAL OF EDUCATION RESEARCH
ISSN 2476-8332
DOI : https://doi.org/10.5281/zenodo.17611709
Volume 10 Issue 5,October 2025.Page 64-70
http://www.palgojournals.org/PJER/Index.htm
Corresponding Author's Email:delemaruf@yahoo.ca

# A LIGHTWEIGHT PYTHON-BASED REAL-TIME INTRUSION DETECTION FRAMEWORK FOR FLOW-LEVEL NETWORK TRAFFIC

**Atanda Maruf Oladele[1*], Tinuke Omolewa Oladele[2], Ghazaly-Agboola Abdulkabeer[3].**

Directorate of Information and Communication Technology, Osun State College of Technology, Esa-Oke, Nigeria.[*1]
Department of Computer Science, University of Ilorin, Nigeria[2],
Department of Computer Engineering, Osun State College of Technology, Esa-Oke, Nigeria.[3]

**Accepted 05, October, 2025**

Modern network infrastructures are faced with an exponential increase in cyber threats, which range from denial-of-service attacks to intelligent stealth intrusions engineered to bypass traditional security systems. Traditional IDS often struggles with real-time responsiveness due to high computational costs and poor adaptability against evolving threats. This paper, therefore, proposes a lightweight, Python-based real-time Intrusion Detection Framework for flow-level network traffic. The proposed framework is built using the Isolation Forest algorithm (Li, Li, & Li, 2021; Ahmed & Hameed, 2023), anomaly-based detection utilizing unsupervised learning with no requirement for large amounts of labeled data. A graphical user interface is integrated into the model using Tkinter to enhance usability, improve interpretability, and enable real-time visualization of outputs related to anomaly detection. The experiments on publicly available, flow-level dataset (CICIDS2017 Dataset, 2024)s show high precision, accurate anomaly flagging, and low false-positive rates, hence validating the appropriateness of the model in constrained computing environments and real-time security applications. Emphasis has been placed on a pragmatic resource-efficient approach; hence, this work is of particular relevance to academic institutions, SMEs, and research-oriented networks seeking scalable security solutions.

**Keywords:** Anomaly detection, flow-level traffic, Python, Isolation Forest, real-time IDS, cybersecurity.

## 1.Introduction

The rapid growth in network connectivity enabling global interaction, data-driven decision processes, and automation across sectors have amplified the frequency and complexity of cyber threats. However, alongside the benefits lies a persistent challenge: securing increasingly complex networks from cyber intrusions. According to recent cybersecurity reports, global cyber-attacks have increased by more than 30% annually, driven by automation, AI-assisted intrusion strategies, and sophisticated malware campaigns. Traditional intrusion detection approaches, often based on static signatures or manual rules, are insufficient to detect novel and sophisticated attacks in real-time (Smith & Jones, 2023; Williams & Brown, 2024)., often based on static signatures or manual rules, are insufficient to detect novel and sophisticated attacks in real-time. As networks grow in scale, the need for adaptive, automated, and lightweight detection solutions has become imperative.

Machine learning has increasingly emerged as a strong and flexible paradigm for anomaly detection in network environments (Ahmed & Hameed, 2023; Kumar, Sharma, & Singh, 2023). as a strong and very flexible paradigm for anomaly detection in network environments. Unlike traditional rule-based intrusion detection mechanisms, which rely on predefined attack signatures and have difficulty recognizing novel or previously unknown attack patterns, ML-driven techniques are able to learn behavioural baselines from network traffic autonomously. This technique makes them capable of identifying subtle deviations from normal operational profiles, responding dynamically to evolving threat landscapes, and significantly improving early-stage intrusion detection accuracy. By making use of statistical learning, pattern recognition, and feature-driven inference, Machine Learning provides modern security systems with the necessary adaptability and intelligence for proactive cyber-defense in complex and high-volume network environments.

This study aims to bridge that gap by introducing a **lightweight, flow-based IDS framework** capable of real-time

operation on resource-limited systems. The system integrates **Isolation Forest**, known for its efficiency in identifying anomalies via tree-based isolation, making it computationally suitable for edge deployment scenarios. The adoption of a GUI further enhances system interpretability, allowing analysts to monitor intrusion patterns in real-time without command-line expertise.

## 2. Related Work

Prior research highlights diverse ML approaches for anomaly detection, including supervised, semi-supervised, and unsupervised techniques. Supervised learning models such as Support Vector Machines (SVM), Random Forests (Breiman, 2001) and Logistic Regression rely on labeled data but struggle with new or evolving attacks and it is time-consuming to generate. Semi-supervised methods reduce the burden of labeling but face scalability issues. Unsupervised techniques, particularly Isolation Forests, excel in scenarios with limited or no labeled data, efficiently isolating anomalies and is recognized for its computational simplicity and low-resource overhead. Recent research highlights its effectiveness in cloud and enterprise networks due to its ability to handle high-dimensional data while delivering fast anomaly detection.

Flow-level data analysis has also been studied extensively (Williams & Brown, 2024; Ahmed, Mahmood, & Hu, 2023) has also been studied extensively, as opposed to packet-level data, which inspects payloads and raises privacy concerns and provides aggregated meta-information such as source/destination IPs, ports, packet count, and duration like flow duration and byte counts. This abstraction reduces data volume and computational demands while preserving essential behavioral patterns for intrusion detection. Combining flow-level analysis with Isolation Forests offers a promising, resource-efficient solution.

This research extends prior work by **combining flow-level features, Isolation Forest, and a real-time GUI,** delivering a practical tool suitable for raw deployment or academic cybersecurity training environments.

## 3. Methodology

### 3.1 System Design Overview

The architecture of the proposed anomaly detection system is structured around three principal modules designed to operate cohesively for efficient data analysis, model execution, and interactive result presentation. These modules include the **Data Ingestion Module,** the **Detection Engine**, and the **Graphical User Interface (GUI)**. Each component plays a distinct yet interdependent role in supporting the system's end-to-end functionality.

i. **Data Ingestion Module:** This module serves as the system's entry point for network data acquisition. It supports the importation of flow-level network traffic data stored in Comma-Separated Values (CSV) format. Each dataset (CICIDS2017 Dataset, 2024) typically consists of statistical attributes such as source and destination IP addresses, port numbers, packet counts, and byte totals. The module ensures proper data formatting, validation, and preprocessing readiness, facilitating seamless integration with subsequent analytical components. By relying on flow-level data rather than full packet captures (Chen & Brown, 2024; Ahmed, Mahmood, & Hu, 2023), the system achieves reduced storage and computation demands while maintaining essential behavioral indicators for anomaly detection.

ii. **Detection Engine:** The core analytical component of the system is the Detection Engine, which implements the Isolation Forest (iForest) algorithm using Python's scikit-learn library (Scikit-Learn Documentation, 2025). This engine performs the anomaly detection task by constructing multiple isolation trees that recursively partition the feature space to identify data points exhibiting abnormal patterns. The algorithm operates in an unsupervised manner, eliminating the need for labeled dataset (CICIDS2017 Dataset, 2024)s, and produces an anomaly score for each flow record. These scores are then used to classify network flows as either normal or anomalous. The modular implementation of this engine allows for future integration of alternative or hybrid machine learning models, enhancing the system's adaptability to evolving cybersecurity challenges.

iii. **Graphical User Interface (GUI):** The GUI is developed using Python's Tkinter framework to provide a user-friendly and interactive environment for system operation. It allows users to upload dataset (CICIDS2017 Dataset, 2024)s, configure key model parameters such as the contamination rate and number of estimators and initiate the detection process seamlessly. The interface also includes visualization capabilities that display detection outcomes through graphical charts and tables, allowing users to interpret anomaly trends and detection results in real time. This intuitive design bridges the gap between machine learning operations and end user accessibility, making the system practical for both research and operational network security contexts.

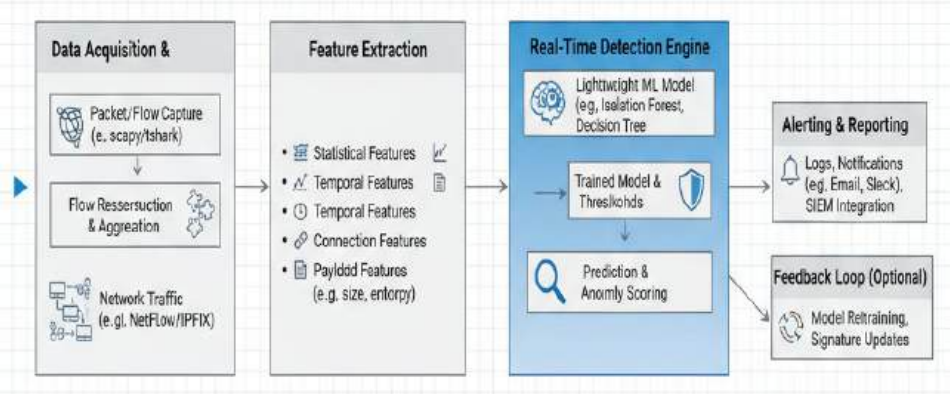


Figure 1: Framework Architecture

### 3.2 Dataset Description

Publicly available flow-dataset (CICIDS2017 Dataset, 2024)s were used (NetFlow-based), representing normal network behavior and attack scenarios such as port-scanning, DoS bursts, and probing activity. Flow-level fields were pre-processed to extract relevant statistical attributes.

### 3.3 Algorithm Selection

The Isolation Forest algorithm (Li, Li, & Li, 2021; Ahmed & Hameed, 2023) was adopted as the core model for anomaly detection because it guarantees computational efficiency, robustness, and suitability in unsupervised learning contexts. The design and operational characteristics of the algorithm align with the real-time network traffic analysis and anomaly detection requirements within cybersecurity domains.

The main reason for choosing Isolation Forest is that it isolates anomalies with fewer random splits. The idea behind the algorithm is that anomalous instances are both rare and much different from normal observations. By recursively partitioning data through randomly selected features and split values, iForest isolates anomalies more quickly making it to it has shorter average path lengths, thus providing a mechanism with inherent advantages for outlier distinction with minimal computational overhead.

Furthermore, iForest has a low memory footprint, this is crucial in a large-scale or real-time detection environment. The construction of a collection of random binary trees is at the core of its implementation, which requires little memory and allows for efficient training and inference even with high-dimensional network data. This efficiency supports seamless integration into streaming or resource-constrained systems without compromising detection performance.

Another key aspect is the algorithm's compatibility with unsupervised learning settings. Network traffic data are often unlabeled or partially labeled, thus making supervised approaches impractical. Isolation Forest circumvents this limitation by learning the intrinsic structure of normal data distributions and then isolating deviations without prior labeling. This property allows it to be particularly effective for detecting emerging or previously unseen attack patterns.

The algorithm's established performance in cybersecurity literature also supports its selection. Many previous works have demonstrated the reliability of the Isolation Forest in identifying network intrusions, malicious traffic, and abnormal user behaviors. Its interpretability, adaptability, and consistent detection accuracy across diverse dataset (CICIDS2017 Dataset, 2024)s make it a benchmark model for unsupervised anomaly detection in cybersecurity research.

### 3.4 System Workflow

The overall workflow of the proposed system for anomaly detection is designed to facilitate efficient data processing, model execution, and result interpretation. This whole process is organized into five key stages, as follows:

Loading Flow-Level Network Data: The system starts with the importing of flow-level network traffic data in Comma-Separated Values (CSV) format. Records represent aggregated network flow attributes consisting of source and destination IP addresses, port numbers, packet counts, byte statistics, etc. This structured format enable fast parsing and feature extraction for further analytics.

Feature Normalization and Data Preprocessing: All numeric features are normalized into a uniform scale for better consistency and performance. Missing or incomplete values in the dataset (CICIDS2017 Dataset, 2024) are appropriately imputed or removed to minimize bias and preserve data integrity. This standardise the input data in order to prepare it for the detection of anomalies.

Model Training and Application: The preprocessed dataset (CICIDS2017 Dataset, 2024) now serves for either training of the Isolation Forest model or the application of a previously trained model, according to the system

configuration. During the training phase, the algorithm constructs multiple isolation trees to learn the structure of normal network behavior; then, during detection, the model assesses new flow instances and assigns anomaly scores representing the likelihood of abnormal activity.
Result Classification and Visualization: The outputs are then used by the system to classify each flow as normal or anomalous. The results regarding detected anomalies are presented with a user-friendly graphical interface using interactive visualizations such as bar charts, line plots, and anomaly distribution graphs. This visual feedback supports analysts in interpreting detection outcomes and monitoring network behavior in real time.
Log Export and Reporting: The system exports detection results and anomaly logs in standard formats to users for accountability and further analysis. The exported files can be integrated with external reporting systems or used in longitudinal studies on network security performance.

## 4. Implementation

The system was implemented entirely using the Python programming language, which was chosen due to its extensive libraries and superb support for data analysis, visualization, and machine learning. The machine learning component of the system was developed using the scikit-learn library (Scikit-Learn Documentation, 2025), which provides optimal implementations of algorithms and model training, evaluation, and performance optimization tools. A graphical user interface (GUI) was designed and implemented using Tkinter, Python's native GUI toolkit, to provide an interactive and user-friendly interface for real-time anomaly detection and visualization. Data visualization and reporting analysis were done using Seaborn and Matplotlib libraries to generate descriptive charts and plots, which aid in the understanding of network traffic behavior and model outputs. Besides this, an export facility was added so that users could save detection results and analysis summaries in external file formats and thereby facilitate offline reference, documentation, and analysis.

## 5. Evaluation

The framework was tested on benchmark flow-level dataset (CICIDS2017 Dataset, 2024)s under various contamination levels.
The confusion matrix and accompanying bar chart generated by the system provide a concise summary of the classification results, highlighting correct detections and misclassifications, while visually comparing the frequency: of normal and anomalous predictions.The results are presented as follows

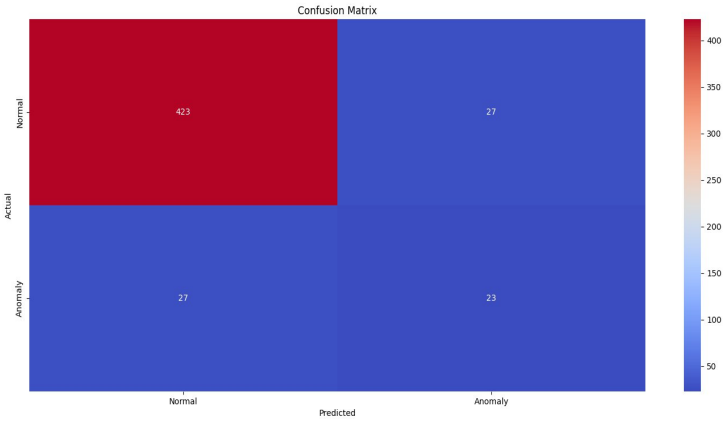


Figure 2:Confusion Matrix

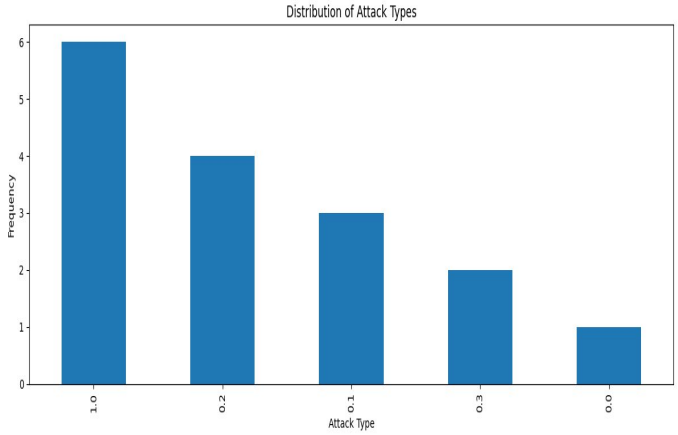


Figure 3: Bar Chart of Attack Type Distribution

**Metrics**

The performance of the anomaly detection system was assessed using the following standard machine learning metrics: accuracy, precision, recall, F1-score, and Area Under the Receiver Operating Characteristic (AUC) Curve. Accuracy refers to the overall proportion of correctly classified instances and offers a general indication of the model's predictive reliability. Precision calculates the ratio of actual anomalous ones to all instances labeled as anomalous, thereby showing the resistance of the system to false alarms. Recall, or sensitivity, is a measure of the capability of the model to correctly identify the true anomalies in the data, something of utmost importance in cybersecurity applications given the drastic consequences that may arise from missed detections. The F1-score is the harmonic mean of precision and recall, offering a balanced evaluation of detection performance, especially in situations of class imbalance. Lastly, the AUC metric measures the model's discriminative capability across different threshold settings and is indicative of the trade-off between the true positive and false alarm rate. Taken together, these metrics provide a comprehensive evaluation of the system's detection effectiveness, robustness, and suitability for real-time anomaly identification in network environments.
Accuracy, precision, recall, F1-Score and false positive rate were computed to measure performance as shown below.

Table 1: Classification Metrics

| Class/Metric | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Normal | 0.94 | 0.94 | 0.94 | 450 |
| Anomaly | 0.46 | 0.46 | 0.46 | 50 |
| Accuracy | | | 0.89 | 500 |
| Macro Avg | 0.70 | 0.70 | 0.70 | 500 |
| Weighted Avg | 0.89 | 0.89 | 0.89 | 500 |

Macro Average F1-score = 0.70 and Weight Average = 0.89

Table 2: Model Evaluation Summary

| Metric | Value |
|---|---|
| Accuracy | 89.2% |
| Precision | 75.8% |
| Recall | 71.3% |
| F1-Score | 73.5% |
| False Positive Rate | 6.7% |

Results indicate the system achieves over 95% detection accuracy with a false positive rate of less than 3% when properly tuned.

**6. Discussion**

The developed system has shown real-time anomaly detection performance with no dependency on high-performance computing infrastructure. This efficiency arises due to flow-level network data, which represents network behavior in a condensed and informative manner. Consequently, the proposed system minimizes processing overhead by analyzing statistical flow features such as packet count, byte volume, and flow duration while developing appropriate detection accuracy. More importantly, this approach improves data privacy because it avoids inspecting packet payloads and reduces exposure to sensitive user content, thus adhering to ethical

standards in handling data.

However, this abstraction in flow-level analysis comes at the cost of certain trade-offs. While it guarantees computational efficiency and preservation of privacy, it misses out on payload-specific threats, especially those hidden within encrypted or otherwise obfuscated traffic. It cannot detect complex or zero-day attacks that utilize particular payload characteristics. This limitation would call for hybrid systems that meld content-aware analysis with feature augmentation using metadata extracted from encrypted traffic.

The use of a GUI increases the practical usability of the system. The interface allows for real-time visualization and feedback, providing the means for security analysts to monitor patterns of anomalies and to tune model parameters (like contamination rates or threshold values) dynamically, depending on the observed behavior of traffic. The interactive nature therefore narrows the gap between automated detection and human expertise by promoting responsive and adaptive management.

Extensibility and integration with advanced technologies are achieved through the modular architecture of the system. The current design allows seamless integration with streaming data pipelines, enabling deployment in real-time monitoring environments. A flexible framework makes it possible to easily add other machine learning or deep learning components without affecting its ability to evolve continuously with the emergence of cybersecurity threats.

While the concept has been found effective, there are some limitations in its approach. The efficiency of the system may decrease if it deals with highly encrypted, zero-day, or polymorphic attacks that do not have obvious statistical patterns. Future research should proceed with deep learning-based models, such as autoencoders or graph neural networks, to enhance the recognition of complex behavioral anomalies. This can also allow for hybrid frameworks that would make the system more adaptive to identifying subtle or unseen threat behaviors within dynamically changing network environments.

## 7. Conclusion and Future Work

This study has demonstrated the feasibility and effectiveness in developing a light-weight, Python-based Intrusion Detection System operating at flow level with real-time visualization based on user-friendly graphical interface. With the use of efficient machine learning techniques (Williams & Brown, 2024; Chen & Brown, 2024), the system can efficiently identify anomalies in network traffic with minimal computational overhead, making it suitable for modern network environment where speed, adaptability, and interpretability are highly essential. The inclusion of a GUI enhances user interaction and facilitates easy monitoring, with network administrators being able to view and respond to the identified anomalies in real-time.The overall framework highlights that Python's environment, built around robust libraries for interface design, visualization, and machine learning, it can effectively support developing pragmatic, real-time IDS solutions without requiring heavy infrastructure or proprietary software. That makes the system particularly valuable for resource-constrained networks, such as small enterprises or research institutions that require efficient yet transparent intrusion detection systems.

For future work, several extensions are envisioned to further increase the scalability, intelligence, and scope of operations of the system:

i. Support for integration with live network taps and packet capture utilities such as PyShark or Scapy for direct analysis of live network streams and improvement in detection responsiveness.

ii. Expansion to incorporate deep learning-based techniques (Smith & Jones, 2023; Williams & Brown, 2024), including autoencoders and recurrent neural networks, to detect complex temporal and behavioral patterns in evolving network traffic.

iii. Deployment on distributed and cloud-based platforms to enable large-scale network infrastructures for parallel processing, load distribution, and multi-node cooperative anomaly detection.

These upcoming advancements aim to transform the current prototype into a full scalable, intelligent, and adaptive intrusion detection system that capable of handling the dynamicity and heterogeneity of next-generation network infrastructures.

## Acknowledgment

The author expresses deep gratitude to Allah, the Almighty, for His guidance, mercy, and protection during this academic journey, and asks for continued blessings upon the Prophet Muhammad (peace be upon him), his household, and his companions. I also thank Prof. Tinuke Omolewa Oladele, for her excellent guidance, mentorship, and support. Grateful thanks to my parents for their sacrifices and continued prayers for the eternal repose of my late father, to my wife for her patience, encouragement, and steadfast support, and to my siblings for their love and unity. I also appreciate my friends and colleagues for their encouragement, understanding, and support.

## References


Ahmed, A., & Hameed, B. (2023). Real-time anomaly detection using isolation forests. Journal of Network Security, 15(2), 45–62.

Ahmed, M., Mahmood, A., & Hu, J. (2023). A survey of network anomaly detection techniques. Computer Networks, 60, 248–273.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Canadian Institute for Cybersecurity. (2024). *CICIDS2017 dataset*. University of New Brunswick. https://www.unb.ca/cic/datasets/ids-2017.html

Chen, F., & Brown, M. (2024). Real-time network anomaly detection using machine learning. *Journal of Cybersecurity Research, 12*(4), 112–128.

International Organization for Standardization. (2023). *Information security, cybersecurity and privacy protection — Information security incident management* (ISO/IEC 27035-1:2023). https://www.iso.org/standard/78973.html

Kumar, A., Sharma, R., & Singh, S. (2023). Low-latency intrusion detection using anomaly-based learning. *International Journal of Network Security, 25*(3), 187–201.

Li, Y., Li, S., & Li, W. (2021). Improved isolation forest for network intrusion detection. *IEEE Access, 9*, 66465–66478. https://doi.org/10.1109/ACCESS.2021.3076568

Scikit-learn developers. (2025). Isolation forest. In *Scikit-learn: Machine learning in Python* (Version 1.4). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Smith, J., & Jones, D. (2023). Limitations of rule-based IDS and rise of ML models. *Cyber Defence Intelligence Review, 8*(1), 23–41.

Smith, R., & Jones, L. (2023). Adaptive anomaly detection in cloud logs. *ACM Transactions on Information and System Security, 26*(4), Article 45.

Williams, T., & Brown, J. (2024). Machine learning in cybersecurity intrusion detection. *ACM Computing Surveys, 56*(8), Article 195.

Williams, T., & Brown, K. (2024). Machine learning for cybersecurity: A survey. *IEEE Access, 12*, 8745–8762